前回まとめ

- リレーショナルデータモデルの**構造記述**
  - 定義に使われている数学的用語
  - リレーション，テーブル
  - スキーマ
  - インスタンス，タップル
  - 第1正規形
  - ~~空値~~
- 次回は第3章

データベース第3回

第3章　リレーショナルデータモデル
―意味記述―

今回の内容

- リレーショナルデータベースの**意味記述**
- 意味：制約
  - キー（主キー），キー制約
  - 外部キー，外部キー制約
  - その他の一貫性制約
- これらは全て，SQLで記述される定義の中で明記される

リレーション中のタップルを識別

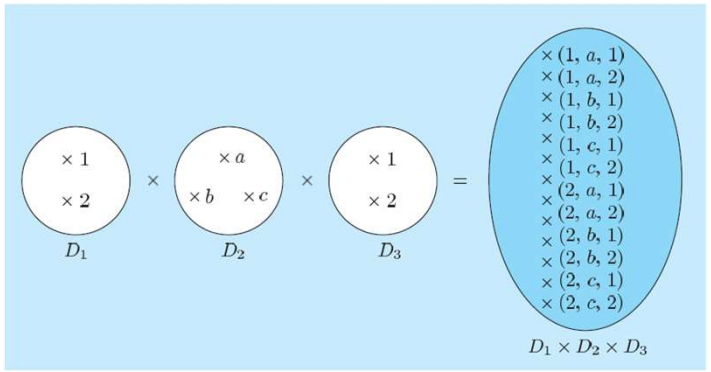
- リレーションの中のタップルを識別するには，どうすれば良い？

学生

学籍番号	学生名	住所
S1	田中	横浜
S2	鈴木	東京
S3	佐藤	横浜

リレーション（数学的用語で定義）

- ドメイン：データの定義域
  - 人名，年令，給与
- 直積
- タップル
- リレーション：ドメインD_1,D_2,・・・,D_n上のリレーションとは，これらドメインの直積$D_1 \times D_2 \times \cdots \times D_n$の任意の有限部分集合
- 濃度（基数）：タップルの総数
- 次数：リレーションが定義されているドメインの数



次数3=3項(ternary)リレーション

次数がnのときは，n項(n-ary)リレーション

リレーションのテーブル表現（前回の講義より）

1	a	1
1	a	2
2	b	1
2	c	2

行(row)：タップル

列
(column)

<u>行の並び順は，何の情報も担っていない</u>
<u>列の順番には意味がある</u>

キー

- リレーションの中のタップルを識別する，つまり，「このタップル」という場合，どうすれば良い？

学生

学籍番号	学生名	住所
S1	田中	横浜
S2	鈴木	東京
S3	佐藤	横浜

リレーション＝有限個のドメインの直積の有限部分集合

行の順番は自由：「何行目」などでは識別できない

⇓
各属性の値を指定する

キー

- リレーションの中のタップルを識別する，つまり，「このタップル」という場合，どうすれば良い？

リレーション＝有限個のドメインの
直積の有限部分集合

学生

学籍番号	学生名	住所
S1	田中	横浜
S2	鈴木	東京
S3	佐藤	横浜

①学籍番号，学生名，住所の値を指定
②学籍番号，学生名の値を指定
③学籍番号，住所の値を指定
④学籍番号を指定

学籍番号（＝極小組の属性）がキー

注：極小とはその属性の組から1個属性を取り除くと
タップルの一意識別能力を失ってしまう，という意味

キー

- リレーション　学生　が以下だったら？

学生

学生名	住所
田中	横浜
鈴木	東京
佐藤	横浜

リレーション＝有限個のドメインの
直積の有限部分集合

①学生名の値を指定→×
②住所の値を指定→×
③学生名，住所の値を指定→〇

一つのリレーションの中に
同じ値のタップルは複数存在しない
∵リレーションは集合だから

候補キーと主キー

社員

社員番号	社員名	給　与	所　属	健保番号
0650	山田太郎	50	K55	80596
1508	鈴木花子	40	K41	81403
0231	田中桃子	60	K41	80201
2034	佐藤一郎	40	K55	81998

キーはどれ？

候補キーと主キー

社員

社員番号	社員名	給　与	所　属	健保番号
0650	山田太郎	50	K55	80596
1508	鈴木花子	40	K41	81403
0231	田中桃子	60	K41	80201
2034	佐藤一郎	40	K55	81998

キーはどれ？

①社員番号
②健保番号

キーとなる属性の組が複数ある場合
それらを候補キーという

その中の一つを主キーという
どれを主キーにするかに決まりはない

```
CREATE SCHEMA 社員–部門 AUTHORIZATION U007;
CREATE DOMAIN 部門番号型 NCHAR(3);
CREATE TABLE 社員 (
        社員番号 INTEGER,
        社員名 NCHAR VARYING(10) NOT NULL,
        給与 INTEGER,
        所属 部門番号型,
        PRIMARY KEY(社員番号),
        FOREIGN KEY(所属) REFERENCES 部門 (部門番号),
        CHECK (20<=
                SELECT AVG(給与)
                FROM 社員),
        CHECK (所属 IN ('K55', 'K41', 'その他')));
CREATE TABLE 部門 (
        部門番号 部門番号型,
        部門名 NCHAR VARYING(10) NOT NULL,
        部門長 INTEGER,
        部員数 INTEGER,
        PRIMARY KEY(部門番号),
        FOREIGN KEY(部門長) REFERENCES 社員 (社員番号));
CREATE VIEW 貧乏社員
        AS SELECT *
        FROM 社員
        WHERE 給与 < 20;
CREATE ASSERTION 給与制約
        CHECK (NOT EXISTS
                (SELECT X.*
                 FROM 社員 X, 社員 Y, 部門 Z
                 WHERE X.所属 = Z.部門番号
                    AND Z.部門長 = Y.社員番号
                    AND X.給与 > Y.給与));
CREATE TRIGGER 部員数整合
        AFTER INSERT ON 社員
            UPDATE 部門
            SET 部員数 = 部員数 + 1
            WHERE 部門.部門番号 = 社員.所属;
```

キー制約

主キーは次の条件を満たさなければならない

1. 主キーはタップルの一意識別能力を備えていること
2. 主キーを構成する属性は空(null)をとらないこと

外部キー

社員

社員番号	社員名	給与	所属	健保番号
0650	鈴木一郎	50	K55	80596
1508	浜崎アユ	40	K41	81403
0231	宇田ひかる	60	K41	80201
2034	別所幸治	40	K55	81998

外部キー

部門

部門番号	部門名	部門長	部員数
K55	データベース	0650	300
K41	ネットワーク	1508	200

複数のテーブルが
あるとき
あるテーブルの属性集合と
外部のテーブルの
主キーの間に関連がある

外部キー

社員

社員番号	社員名	給与	所属	健保番号
0650	鈴木一郎	50	K55	80596
1508	浜崎アユ	40	K41	81403
0231	宇田ひかる	60	K41	80201
2034	別所幸治	40	K55	81998

外部キー

部門

部門番号	部門名	部門長	部員数
K55	データベース	0650	300
K41	ネットワーク	1508	200

リレーション　社員の
所属　には，
リレーション　部門に
登録されている部門番号か
空しか入り得ない

リレーション　部門の
部門長　には，
リレーション　社員に
登録されている社員番号か
空しか入り得ない

外部キー制約

リレーションR(・・・, A_i,・・・)の属性 A_i がリレーションS($\underline{B_1}$,・・・)に関する外部キーであるならば，Rの任意のタップル t に対して，$t[A_i]$ は空であるか，そうでない場合にはSにあるタップル u が存在して，$t[A_i]=u[B_1]$ でなければならない．ここに，$t[A_i]$ と $u[B_1]$ はそれぞれ t と u の A_i 値と B_1 値を表す．

外部キー

社員

社員番号	社員名	給与	所属	健保番号
0650	鈴木一郎	50	K55	80596
1508	浜崎アユ	40	K41	81403
0231	宇田ひかる	60	K41	80201
2034	別所幸治	40	K55	81998

外部キー

リレーション 社員の所属 には，リレーション 部門に登録されている部門番号か空しか入り得ない

部門

部門番号	部門名	部門長	部員数
K55	データベース	0650	300
K41	ネットワーク	1508	200

リレーション 部門の部門長 には，リレーション 社員に登録されている社員番号か空しか入り得ない


```
CREATE SCHEMA 社員–部門 AUTHORIZATION U007;
CREATE DOMAIN 部門番号型 NCHAR(3);
CREATE TABLE 社員 (
        社員番号 INTEGER,
        社員名 NCHAR VARYING(10) NOT NULL,
        給与 INTEGER,
        所属 部門番号型,
        PRIMARY KEY(社員番号),
        FOREIGN KEY(所属) REFERENCES 部門 (部門番号),
        CHECK (20<=
                SELECT AVG(給与)
                FROM 社員),
        CHECK (所属 IN ('K55', 'K41', 'その他 ')));
CREATE TABLE 部門 (
        部門番号 部門番号型,
        部門名 NCHAR VARYING(10) NOT NULL,
        部門長 INTEGER,
        部員数 INTEGER,
        PRIMARY KEY(部門番号),
        FOREIGN KEY(部門長) REFERENCES 社員 (社員番号));
CREATE VIEW 若之社員
        AS SELECT *
        FROM 社員
        WHERE 給与 < 20;
CREATE ASSERTION 給与制約
        CHECK (NOT EXISTS
                (SELECT X.*
                 FROM 社員 X, 社員 Y, 部門 Z
                 WHERE X.所属 = Z.部門番号
                    AND Z.部門長 = Y.社員番号
                    AND X.給与 > Y.給与));
CREATE TRIGGER 部員数整合
        AFTER INSERT ON 社員
            UPDATE 部門
            SET 部員数 = 部員数 +1
            WHERE 部門.部門番号 = 社員.所属;
```

その他の一貫性制約：検査制約

- SQLでCHECKにより記述
  - 社員の所属値は必ず「K55」か「K41」か「その他」でなければならない
  - 年令は16以上70以下でなければならない
  - 社員の平均給与は20以上でなければならない

```
CREATE TABLE 社員 (
        ⋮
        所属 NCHAR(3),
        CHECK(所属 IN ('K55', 'K41', 'その他 ')),
        ⋮
        年齢 INTEGER,
        CHECK( 年齢 >= 16 ),
        ⋮
        給与 INTEGER,
        CHECK( 20 <=
                (SELECT AVG(給与)
                 FROM 社員) ),
        ⋮
)
```

その他の一貫性制約：表明

- SQLでASSERTIONにより記述
  - 上司よりも高給をとっている社員がいてはいけない

```
CREATE ASSERTION 給与制約
    CHECK (NOT EXISTS
            (SELECT X.*
             FROM   社員 X, 社員 Y, 部門 Z
             WHERE  X.所属 = Z.部門番号
               AND  Z.部門長 = Y.社員番号
               AND  X.給与 > Y.給与))
```


その他の一貫性制約：トリガ

- トリガ(trigger)：引き金，きっかけ
  - 社員ー部門 データベースで，リレーション 社員 に新入社員の挿入（や会社を辞めた人の削除）に対応して，リレーション 部門 の部員数を増（減）しなければならない

社員

社員番号	社員名	給与	所属	健保番号
0650	鈴木一郎	50	K55	80596
1508	浜崎アユ	40	K41	81403
0231	宇田ひかる	60	K41	80201
2034	別所幸治	40	K55	81998

部門

部門番号	部門名	部門長	部員数
K55	データベース	0650	300
K41	ネットワーク	1508	200

その他の一貫性制約：トリガ

- トリガ(trigger)：引き金，きっかけ
  - 社員ー部門 データベースで，リレーション 社員 に新入社員の挿入（や会社を辞めた人の削除）に対応して，リレーション 部門 の部員数を増（減）しなければならない

```
CREATE TRIGGER 部員数整合
    AFTER INSERT ON 社員
        UPDATE 部門
        SET 部員数 = 部員数 + 1
        WHERE 部門.部門番号 = 社員.所属
```

図 3.6　トリガの定義例


```
CREATE SCHEMA 社員-部門 AUTHORIZATION U007;
CREATE DOMAIN 部門番号型 NCHAR(3);
CREATE TABLE 社員 (
        社員番号 INTEGER,
        社員名 NCHAR VARYING(10) NOT NULL,
        給与 INTEGER,
        所属 部門番号型,
        PRIMARY KEY(社員番号),
        FOREIGN KEY(所属) REFERENCES 部門 (部門番号),
        CHECK (20<=
                SELECT AVG(給与)
                FROM 社員),
        CHECK (所属 IN ('K55','K41','その他')));
CREATE TABLE 部門 (
        部門番号 部門番号型,
        部門名 NCHAR VARYING(10) NOT NULL,
        部門長 INTEGER,
        部員数 INTEGER,
        PRIMARY KEY(部門番号),
        FOREIGN KEY(部門長) REFERENCES 社員 (社員番号));
CREATE VIEW 貧乏社員
        AS SELECT *
        FROM 社員
        WHERE 給与 < 20;
CREATE ASSERTION 給与制約
        CHECK (NOT EXISTS
                (SELECT X.*
                 FROM 社員 X, 社員 Y, 部門 Z
                 WHERE X.所属 = Z.部門番号
                   AND Z.部門長 = Y.社員番号
                   AND X.給与 > Y.給与));
CREATE TRIGGER 部員数整合
        AFTER INSERT ON 社員
                UPDATE 部門
                SET 部員数 = 部員数 + 1
                WHERE 部門.部門番号 = 社員.所属;
```

リレーショナルデータベーススキーマ

- リレーションスキーマ＋一貫性制約
  ＝データベーススキーマ

まとめ

- リレーショナルデータベースの**意味記述**
- 意味：制約
  - キー（主キー），キー制約
  - 外部キー，外部キー制約
  - その他の一貫性制約